**Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**UCS2612 Machine Learning Laboratory**

**Assignment 5**

**Name:** V.S. Pranav **Reg No:** 3122 21 5001 069

**k-Nearest Neighbor algorithm**

**Github Link:** <https://github.com/vspr14/ml-lab-assn5>

1. Develop a python program to predict the Online Shoppers Purchasing Intention using K-Nearest Neighbour algorithm. Visualize the features from the dataset and interpret the results obtained by the model using Matplotlib library.

**Code:**

*# %%*

*import* pandas *as* pd

*from* imblearn.over\_sampling *import* SMOTE

*from* imblearn.under\_sampling *import* RandomUnderSampler

*from* sklearn.model\_selection *import* train\_test\_split

*import* numpy *as* np

*from* sklearn *import* preprocessing

*from* sklearn.metrics *import* accuracy\_score, roc\_auc\_score, roc\_curve, confusion\_matrix,f1\_score, precision\_score, recall\_score

*from* scikitplot.metrics *import* plot\_confusion\_matrix

*# %%*

df = pd.read\_csv("online\_shoppers\_intention.csv")

label\_encoder = preprocessing.LabelEncoder()

*#Preprocessing*

df['Month']= label\_encoder.fit\_transform(df['Month'])

df['VisitorType']= label\_encoder.fit\_transform(df['VisitorType'])

df['Weekend']= label\_encoder.fit\_transform(df['Weekend'])

df['Revenue']= label\_encoder.fit\_transform(df['Revenue'])

*# %%*

*# Creating an instance of SMOTE*

X = df.drop("VisitorType", *axis*=1)

correlation = df.corr()

correlation = correlation['Revenue'].sort\_values()

correlation = correlation[correlation > 0.1]

columns = list(correlation.index)[:-1]

*# X = df[columns]*

X = df[["PageValues","Month","ExitRates"]]

y = df["Revenue"]

smote = SMOTE()

rus = RandomUnderSampler(*random\_state*=42, *sampling\_strategy* = 'majority')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, *test\_size*=0.2, *random\_state*=42)

*# %%*

print((y.value\_counts()))

X\_resampled, y\_resampled = smote.fit\_resample(X\_train, y\_train)

print((y\_resampled.value\_counts()))

*# %%*

*from* sklearn.neighbors *import* KNeighborsClassifier

model = KNeighborsClassifier(*n\_neighbors*=11)

model2 = KNeighborsClassifier(*n\_neighbors*=11)

model.fit(X\_train, y\_train)

model2.fit(X\_resampled, y\_resampled)

y\_pred = model.predict(X\_test)

y\_pred2 = model2.predict(X\_test)

*# %%*

*import* scikitplot *as* skplt

*import* matplotlib.pyplot *as* plt

y\_pred\_proba = model.predict\_proba(X\_test)[::,1]

auc = roc\_auc\_score(y\_test, y\_pred\_proba)

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba)

plt.plot(fpr,tpr,*label*="with oversampling, auc="+str(auc), )

x = [0, 1]

y = [0, 1]

y\_pred\_proba2 = model2.predict\_proba(X\_test)[::,1]

auc2 = roc\_auc\_score(y\_test, y\_pred\_proba2)

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba2)

print("Accuracy score without oversampling:",accuracy\_score(y\_test, y\_pred))

print("F1 score without oversampling:",f1\_score(y\_test, y\_pred))

print("Precision without oversampling:",precision\_score(y\_test, y\_pred))

print("Recall without oversampling:",recall\_score(y\_test, y\_pred))

print()

print("Accuracy score with oversampling:",accuracy\_score(y\_test, y\_pred2))

print("F1 score with oversampling:",f1\_score(y\_test, y\_pred2))

print("Precision with oversampling:",precision\_score(y\_test, y\_pred2))

print("Recall with oversampling:",recall\_score(y\_test, y\_pred2))

plt.plot(fpr,tpr,*label*="without oversampling, auc="+str(auc2), *color*='red')

plt.plot(x,y)

plt.legend(*loc*=4)

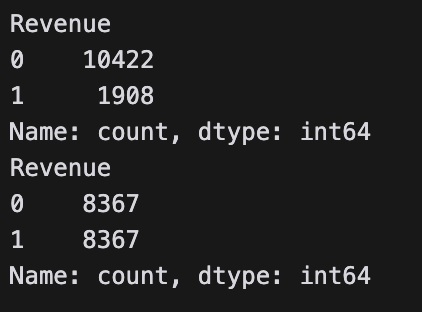
plot\_confusion\_matrix(y\_test, y\_pred, *title*="Without oversampling")

plot\_confusion\_matrix(y\_test, y\_pred2, *title*="With oversampling")

plt.show()

**Sample Screenshots:**

A screen shot of a computer

Description automatically generated

A graph of a number and a line

Description automatically generated with medium confidence

A blue and white graph with numbers and a blue square

Description automatically generated

